Un algorithme d'exclusion mutuelle pour une structure logique en anneau

Michel Raynal

INRIA

CENTRE DE RENNES
IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 350

UN ALGORITHME D'EXCLUSION MUTUELLE POUR UNE STRUCTURE LOGIQUE EN ANNEAU

Michel RAYNAL

Décembre 1984

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

UN ALGORITHME D'EXCLUSION MUTUELLE POUR UNE STRUCTURE LOGIQUE EN ANNEAU

Michel RAYNAL
IRISA - Université de Rennes I
Campus de Beaulieu
35042 RENNES CEDEX

Publication Interne n° 241 - Novembre 1984
12 pages

Résumé. - Nous présentons un nouvel algorithme d'exclusion mutuelle pour un ensemble de processus connectés selon une structure d'anneau virtuel. Cet algorithme basé sur des variables d'états se différencie de celui proposé par DIJKSTRA (C.ACM.74) par le fait qu'il ne présente pas de processus jouant un rôle privilégié: le protocole d'entrée et de sortie de section critique est le même pour tous. Outre cette propriété, l'algorithme en possède deux autres: il ne nécessite que des variables de taille bornée et résiste aux pannes des processus placés sur l'anneau.

ABSTRACT. - A new mutual exclusion algorithm designed for a set of processes connected along a virtual ring is presented. This algorithm is based on state variables and differs from the Dijkstra's one (Comm. ACM 74) by not having a tagged process playing a special part : entry and exit protocols are identical for all the processes. Moreover the algorithm possesses two other properties : (i) it needs only bounded variables and (ii) copes with failures of processes.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

PI 228 **Un nouveau réseau d'interconnexion adapté aux calculateurs SIMD**
André Seznec, 42 pages ; Juillet 1984.

PI 229 **Analyse factorielle en référence à un modèle. Application à l'analyse de tableaux d'échanges**
Brigitte Escofier, 19 pages ; Juillet 1984.

PI 230 **Infinitary languages and fully abstract models of fair asynchrony**
Philippe Darondeau, 79 pages ; Juillet 1984.

PI 231 **Contribution d'une approche syntaxique dans la segmentation d'image**
Jean Camillerapp, Ivan Leplumey, 46 pages ; Juillet 1984.

PI 232 **Analyse d'un algorithme de classification hiérarchique «en parallèle» pour le traitement de gros ensembles**
Israël-César Lerman, Philippe Peter, 114 pages ; Août 1984.

PI 233 **Manuel d'utilisation de Diastol - Version Préliminaire**
Patrice Quinton, Pierrick Gachet, 29 pages ; Août 1984.

PI 234 **Architectures systoliques pour la reconnaissance de mots connectés (en anglais)**
François Charot, Patrice Frison, Patrice Quinton, 40 pages ; Août 1984.

PI 235 **A scheme of Token Tracker**
Zhao Jing Lu, 62 pages ; Septembre 1984.

PI 236 **Design of one-step and multistep adaptive algorithms for the tracking of time varying systems**
Albert Benveniste, 40 pages ; Septembre 1984.

PI 237 **The design and building of Enchère, a distributed electronic marketing system**
Jean-Pierre Banatre, Michel Banatre, Guy Lapalme, Florimond Ployette, 38 pages ; Septembre 1984.

PI 238 **Algorithme optimal de décision pour l'équivalence des grammaires simples**
Didier Caucal, 48 pages ; Septembre 1984.

PI 239 **Detection and diagnosis of abrupt changes in modal characteristics of nonstationary digital signals**
Michèle Basseville, Albert Benveniste, Georges Moustakides, 26 pages ; Octobre 1984.

PI 240 **Convergence optimale de l'algorithme de «réallocation-recentrage» *dans le cas le plus pur***
Israël-César Lerman, 39 pages ; Octobre 1984.

PI 241 **Un algorithme d'exclusion mutuelle pour une structure logique en anneau**
Michel Raynal, 12 pages ; Novembre 1984.

UN ALGORITHME D'EXCLUSION MUTUELLE POUR UNE STRUCTURE LOGIQUE EN ANNEAU

Michel RAYNAL

Publication n° 241

Novembre 1984

TITRE : Un algorithme d'Exclusion Mutuelle pour une Structure Logique en Anneau.

AUTEUR : Michel RAYNAL
IRISA - Université de Rennes - 35042 RENNES CEDEX

DOMAINES: Recherche : Algorithmes Distribués
Génie Logiciel

MOTS CLES: Algorithmes Distribués, Exclusion Mutuelle, Anneau virtuel, Symétrie, Synchronisation, Résistance aux pannes.

TABLE DES MATIERES

GENRE : Courte communication

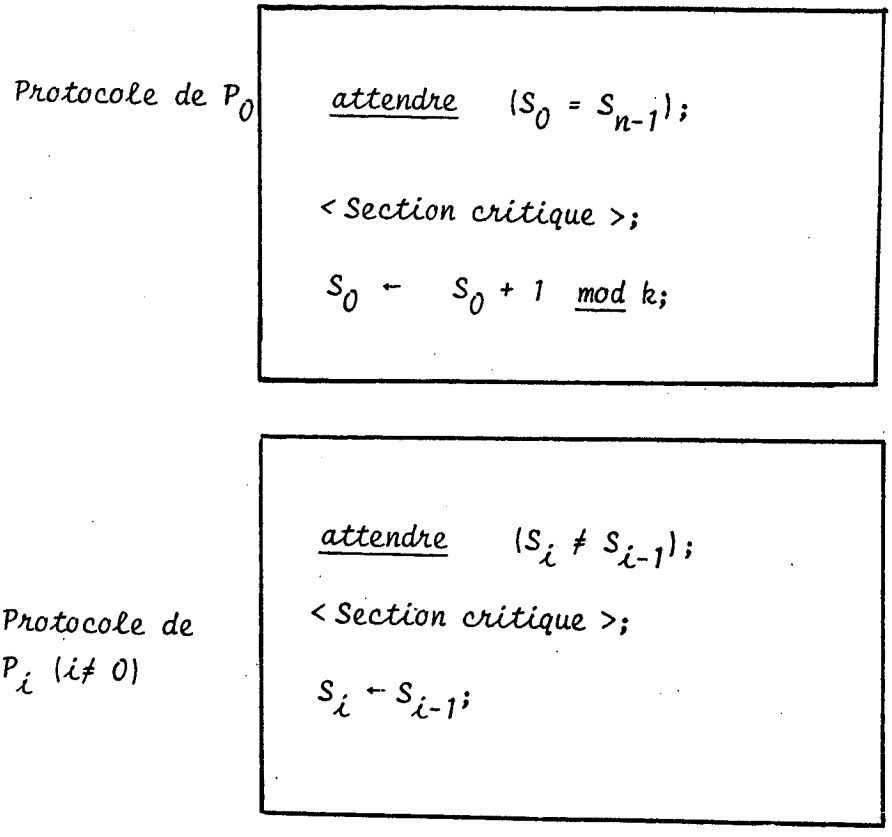
I - INTRODUCTION : L'ALGORITHME DE DIJKSTRA

Dans [Dijkstra74] DIJKSTRA présente un algorithme d'exclusion mutuelle pour un ensemble de processus logiquement connectés selon une structure en anneau; en d'autres termes chaque processus ne connait que ses deux voisins immédiats. En conséquence, le privilège d'être en section critique va tourner sur l'anneau: un processus qui vient d'obtenir le privilège ne l'obtiendra à nouveau que lorsque tous les autres processus l'auront également obtenu. La structure logique d'anneau assure ainsi l'équité et définit la borne maximum de l'attente lorsque tous les processus veulent pénétrer en section critique; si **n** est le nombre de processus la borne est **a(n)=n-1** tours (où un tour est une possession par un processus de la section critique). La contrepartie de la structure en anneau est que lorsqu'un processus possède le privilège et ne désire pas pénétrer en section critique, il doit l'abandonner à son voisin.

L'algorithme de DIJKSTRA ne repose sur aucune variable partagée que tous les processus peuvent lire et écrire comme c'est le cas dans les solutions "traditionnelles" où l'on fait l'hypothèse que tous les processus accèdent en lecture et en écriture une mémoire commune support des variables partagées [Djikstra65, Peterson81, Burns81, Kowaltowski81]; la panne d'une telle variable, qui confère son caractère centralisé à l'algorithme, lui est alors fatale. Dans l'algorithme de DIJKSTRA par contre chaque processus P_i est doté d'une variable S_i, permettant de caractériser son état, distribuée au sens suivant: seul P_i a le droit de lire et d'écrire S_i et seul son voisin de droite sur l'anneau $P_{(i+1 \bmod n)}$, parmi tous les autres processus, peut lir S_i. Nous avons donc dans chacun des processus P_i ($i \in 0..n-1$) la déclaration:

Var distribuée-à-droite S_i : 0..k-1 **initialisée-à** 0 ;

L'algorithme proposé par DIJKSTRA présente des protocoles différents pour les processus P_0 et les autres processus P_i ($i \in 1..n-1$). Ce sont :



On a souvent comparé la transmission du privilège dans cet algorithme à la progression d'un "front" (**mod k**) de P_0 vers P_{n-1}: le processus où se situe le front possédant le privilège et P_0 jouant le rôle de régénérateur du front lorsque celui-ci, parvenu à P_{n-1}, disparait [Cornafion81].

DIJKSTRA présente cet algorithme en insistant sur son caractère auto-stabilisateur: si **k>n**, quelle que soit l'initialisation des variables d'états S_i, au bout d'un nombre fini de transitions entre états, un seul processus possèdera le privilège d'être en section critique. Une démonstration et un calcul de cette borne sont proposés dans [Mossiere77]. Cette propriété académique parait intéressante mais comme le note néanmoins DIJKSTRA, ce n'est pas l'objet de son papier de montrer "whether the property of self stabilization is interesting as a starting procedure for the sake of robustness or merely as an intriguing problem".

Indépendamment donc de l'auto-stabilisation cet algorithme présente une propriété intéressante: les variables d'états sont de taille bornée: **O..k-1.** (Du point de vue de cette propriété il est intéressant de noter la position foncièrement différente d'autres algorithmes d'exclusion mutuelle également construits sur des variables distribuées mais utilisant un mécanisme de tickets ou d'estampilles [Lamport74, Hehner81] qui, s'ils n'obligent pas le privilège à circuler entre tous les processus, nécessitent des variables de taille non bornée a priori).

Un inconvénient de cet algorithme est le caractère dissymétrique du processus P_0. Cet inconvénient n'a pas trait seulement à l'esthétique de la solution mais également à la résistance aux pannes. En effet, lorsqu'un processus $P_i(i\neq 0)$ disparait, si le système reconfigure l'anneau virtuel, l'algorithme n'est pas modifié; par contre la panne de P_0 est fatale à l'algorithme: aucune processus n'étant capable de régénérer le front, [Cornafion81] présente une tentative de solution en offrant les protocoles de P_0 et $P_i(i\neq 0)$ à tous les processus et, selon le jeu des pannes, seul le processus P_k dont le voisin de gauche P_l possède un indice supérieur $l>k$, joue le rôle dissymétrique de P_0.

Nous présentons dans la suite de cet article un nouvel algorithme d'exclusion mutuelle pour un ensemble de processus connectés selon une structure virtuelle d'anneau tel que :

-> le privilège tourne entre les processus,
-> les variables d'état sont distribuées et de taille bornée,
-> tous les processus obéissent au même protocole,
-> les pannes n'altèrent pas le fonctionnement.

II - L'ALGORITHME

Nous considérons **n** processus P_1, P_2, ... P_i,...P_n connectés en anneau, chacun pouvant lire la variable d'état de son voisin de gauche. Les variables d'état ont une taille bornée et prennent leurs valeurs dans $O..n^2-1$. Nous avons ainsi dans chaque processus P_i la déclaration et l'initialisation suivantes

var distribuée-à-droite Si : O .. n^2-1 initialisée-à i ;

Dans l'algorithme seule l'initialisation est distincte pour chacun des processus. Le texte des protocoles restant identique pour tous les processus, la solution satisfait les critères de symétrie définis et utilisés par BURNS dans [Burns81].

Le protocole suivi par chacun des processus P_i est exprimé de la même manière que précédemment: le prélude est une attente (éventuellement active) d'une condition portant sur les variables d'état de P_i et de P_{i-1} et le postlude consiste en la modification de la variable d'état de P_i ce qui contribue à transmettre le privilège au voisin de droite P_{i+1}.

```
attendre   S(i-1) = S(i) + n-1   mod n^2 ;
<Section critique> ;
S(i) ← S(i-1) + 1   mod n^2 ;
```

Comme on le voit au début seul P_i a sa condition vérifiée; il pénètre donc en section critique et l'exécution du postlude positionne S_i à n+1 ce qui valide la condition de P_2 etc... Lorsque P_n obtient sa première exclusion il positionne S_n à 2n ce qui validera la seconde entrée en section critique de P_1 et ainsi de suite. Si l'on porte sur un tableau les valeurs successives prises par les variables d'états nous obtenons:

Processus / tour sur l'anneau	P_1	P_2	...	P_i	P_{i+1}	...	P_{n-1}	P_n
initialisation	1	2		i	$i+1$		$n-1$	n
1	$n+1$	$n+2$		$n+i$	$n+i+1$		$n+n-1$	$2n$
	.	.		.	.		.	.
	.	.		.	.		.	.
	.	.		.	.		.	.
j	$jn+1$	$jn+2$		$jn+i$	$jn+i+1$		$jn+n-1$	$(j+1)n$
	.	.		.	.		.	.
	.	.		.	.		.	.
	.	.		.	.		.	.
$(n-1)$	$(n-1)n+1$	$(n-1)n+2$		$(n-1)n+i$	$(n-1)n+i+1$		$(n-1)n+n-1$	0
n	1	2		i	$i+1$		$n-1$	n

Comme on le voit les valeurs prises par la variable d'état d'un processus sont distinctes de celles prises par les variables de tous les autres; $S_i (i \in 1..n)$ est la seule variable à prendre les valeurs: $jn+i$ avec $j \in 0..n-1$. Ceci explique le choix de n^2 dans **mod** n^2. Borner les variables S_i d'un ensemble de processus au comportement répétitif nécessite de passer par une séquence répétitive de configurations de ces variables. Comme par construction de l'algorithme, chaque variable S_i prend des valeurs distinctes des autres S_j $(j \neq i)$, la donnée de n processus dont chaque variable S_i peut prendre n valeurs distinctes définit la valeur du **mod** n^2.

De même que précédemment on a fait une analogie entre l'algorithme de DIJKSTRA et la progression d'un front, nous pouvons comparer cet algorithme à la montée d'un escalier en relais par n processus: chacun d'eux est sur une marche distincte et contigüe aux autres et le seul qui est autorisé à monter est celui qui a (n-1) marches d'écart avec son voisin de gauche (la relation de voisinage **modulo** n): il s'agit du processus placé sur la marche la plus basse qui prend le relais et monte se placer en tête.

Cet algorithme obéit donc à trois des quatre propriétés énoncées que nous rappelons: le privilège tourne entre tous les processus, les variables d'état sont distribuées et bornées, et le protocole est le même pour tous les processus. Nous allons maintenant examiner le dernier point: le comportement en cas de pannes.

III - LE COMPORTEMENT EN CAS DE PANNE

Lorsqu'un processus tombe en panne, ne pouvant tester la valeur de la variable d'état de son voisin, il ne peut obtenir le privilège, interdisant ainsi sa transmission vers les autres processus de l'anneau: ce dernier est donc considéré comme étant sectionné. Comme dans l'algorithme de DIJKSTRA résoudre le problème de la panne nécessite donc de savoir reconfigurer en une structure d'anneau virtuel.

Pour cela nous introduisons deux variables d'état par processus qui matérialisent explicitement l'anneau virtuel, ces variables étaient implicites dans la formulation précédente. Ce sont :

var k_i : O .. n - 1 **initialisée-à** O ;

VG_i : 1 .. n **initialisée-à**

si i=1 **alors** n **sinon** i-1 **fsi** ;

VG_i et k_i désignent respectivement le voisin de gauche de P_i sur l'anneau et le nombre de processus tombés en panne entre P_{VG_i} et P_i.

Ces variables sont gérées par le logiciel fournissant la structure d'anneau virtuel (noyau du système, couche de transport du réseau par exemple). Avec ces variables l'algorithme prend une forme plus générale tout en conservant les propriétés précédentes (privilège tournant, variables bornées, protocole identique pour tous) auxquelles vient s'ajouter la résistance aux pannes qui lui sont signalées. Le protocole pour P_i ($i \in 1 .. n$) est obtenu en substituant $S_{VG_i} + k_i$ à S_{i-1}, ce qui donne :

```
attendre  S_VG_i + k_i = S_i + n - 1  mod  n^2 ;
< Section critique > ;
S_i ← S_VG_i + k_i + 1  mod  n^2 ;
```

Les mécanismes proposés dans [Mossiere77, Cornafion81], peuvent être utilisés pour signaler les pannes; à cause de leur taille, ils ne sont pas repris ici pour ne pas alourdir l'exposé. Le lecteur interessé trouvera toutefois dans [Lelann77, Lelann78] une présentation de protocoles de gestion d'anneau virtuel et de reconfiguration en cas de pannes (notamment le protocole de suspiscion mutuelle entre les processus). La réinsertion d'un processus peut utiliser de tels mécanismes (qui remettent à jour les variables VG_x et k_x du processus réinséré et de son voisin de droite).

IV - REMARQUES ET CONCLUSION

Une première remarque qui vient à l'esprit est la suivante. Si l'on considère à la fois l'algorithme d'exclusion mutuelle proposé et le mécanisme qui lui signale les pannes l'on peut obtenir un algorithme d'exclusion mutuelle au comportement équitable qui n'oblige pas un processus à prendre le privilège s'il ne le désire pas: il se met alors en panne et ne se réinsère que lorsqu'il désire pénétrer en exclusion; il doit alors attendre au maximum **n-1** tours.

Une seconde remarque concerne la conception des algorithmes où le privilège tourne sur un anneau. Si l'on autorise à tout processus de lire les variables d'état de tous les autres, un algorithme simple d'exclusion mutuelle avec privilège tournant peut être formulé. Dans ce cas les variables d'états dont la taille est bornée par **n** sont initialisées à des valeurs quelconques. Le protocole, symétrique pour tous les processus, est alors pour P_i ($i \in 0 .. n-1$) :

```
attendre   S1 + S2 + ... + Sn = i  mod n;

<Section critique>;

Si ← Si + 1  mod  n;
```

Il y a alors initialement un seul processus en section critique quelque soit l'intialisation. Comme on le voit malgré l'initialisation quelconque cet algorithme ne nécessite pas d'autostabilisation. Si P_i est le premier à posséder l'exclusion, le suivant sera P_{i+1} et ainsi de suite.

On constatera que si le postlude décrémentait S_i le privilège tournerait dans le sens inverse. De manière plus générale si le postlude avait été :

$$S_i \leftarrow S_i + k \bmod n$$

le privilège aurait tourné sur les processus: P_i, P_{i+k}, P_{i+2k} ... Il faut alors que k et n soient premiers entre eux afin qu'aucun processus ne soit définitivement écarté de l'accès en section critique, le privilège tournant sur un sous-anneau virtuel.

Le privilège peut ainsi tourner sur un anneau virtuel indépendant de celui que forment les processus.

Pour conclure, rappelons simplement que l'algorithme que nous avons présenté dans ce papier est intéressant à plusieurs titres. Tout d'abord pour les propriétés qu'il présente: privilège tournant, pas de processus privilégié, variables bornées et résistance aux pannes. Ensuite, par son côté didactique: la symétrie du protocole pour tous les processus rend sa comparaison avec d'autres algorithmes intéressantes et notamment celui de DIJKSTRA que nous avons présenté [Dijkstra74], celui de BURNS qui repose sur des variables partagées [Burns81], et celui de LAMPORT qui utilise des variables non bornées [Lamport74].

Cet algorithme comme ceux de [Dijkstra74, Lamport74, Hehner81] est distribué et basé sur des variables d'état et non sur des messages. Il peut donc s'avérer intéressant pour distribuer le contrôle sur une architecture centralisée. On peut ainsi l'utiliser lorsque un privilège doit tourner sur un ensemble d'entités que ce soit au niveau d'un système opératoire ou d'une application (simulation distribuée, controle de cohérence dans une base de données distribuée, etc.). Signalons enfin au lecteur intéressé qu'il trouvera dans [Raynal84] une présentation synthétique de nombreux algorithmes d'exclusion mutuelle.

REFERENCES

[Burns 81] J.E. Burns : **Symmetry in Systems of Asynchronous Processes ;** 22nd Annual Symp. on fundations of Computer Science, (Oct.81), pp.169-194.

[Cornafion 81] Cornafion (Nom collectif) : **Systèmes Informatiques répartis ;** DUNOD (1981), 368p.

[Dijkstra 65] E.W. Dijkstra : **Solution of a problem in Concurrent Programming Control ;** Comm. ACM, Vol.8,9 (Sept.65), p.569.

[Dijkstra 74] E.W. Dijkstra : **Self-stabilizing systems in spite of distributed Control ;** Comm. ACM, Vol.17, 11 (Nov.74), pp.643-644.

[Hehner 81] E.C.R. Hehner, R.K. Shyamasundar : **An implementation of P and V ;** Inf. Proc. Letters, Vol.12,4, (June 81), pp.196-198.

[Kowaltowski 84] T. Kowaltowski, A. Palma : **Another Solution of the Mutual Exclusion Problem ;** Inf. Proc. Letters, 19 (1984), pp.145-146.

[Lamport 74] L. Lamport : **A new solution of DIJKSTRA's Concurrent Programming Problem ;** Comm. ACM, Vol.17,8 (Aug.74), pp.453-455.

[Lelann 77] G. Lelann : **Distributed systems : towards a formal approach ;** IFIP Congress, Toronto (August, 1977).

[Lelann 78] G. Lelann : **Algorithms for distributed data-sharing systems which use tickets ;** 3rd workshop on distributed data management and computer networks, Berkeley, (August, 1978).

[Raynal 84] M. Raynal : **Algorithmique du parallélisme : le problème de l'exclusion mutuelle ;** DUNOD (1984), 164p.

[Peterson 81] G.L. Peterson : **Myths about the Mutual Exclusion Problem ;** Inf. Proc. Letters, Vol.12,3, (June 81), pp.115-116.

[Mossière 77] J. Mossière, M. Tchuente, J.P. Verjus : **Sur l'exclusion Mutuelle dans les réseaux informatiques ;** Rapport de Recherche n°75, IRISA - RENNES (Oct. 77), 19p.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique